Overview

The Program Halt class handles unexpected run-time errors that would otherwise lead to a program halt in SAS/AF software. By adding a program halt component to your application, you can prevent the end user from seeing program halt errors written to the SAS Log. Instead, the program halt information is stored to an SCL list which enables you to trap and process the information as necessary. You can also use a program halt component to handle different program halts based on the type of error that occurred.

Parent:
sashelp.fsp.Object.class
Class:
sashelp.classes.Programhalt.class

Using the Program Halt Component

The Program Halt Component has methods that are called when certain run-time exceptions occur. To make effective use of a program halt component in your application, you can override these methods to control how the exceptions are handled.

For example, you can create a subclass of Program Halt that overrides the _onZeroDivide method:

```sas
/* error.class overrides programHalt.class */ CLASS sasuser.myclass.error.class extends sashelp.classes.ProgramHalt.class; _onZeroDivide:method / (STATE='O'); /* processing goes here */ endmethod;
```

The _onZeroDivide method overrides the method on the parent class and is invoked when a program halt occurs and the ZeroDivide event is sent.
The SCL code that would cause the program halt and send the ZeroDivide event could look like:

```sas
import sasuser.myclass.error.class; init: dcl num x y; dcl error_errorobj = _new_ error(); return; main: x=0; y=3/x; return;
```

A program halt component enables you to determine whether an application should continue executing or immediately halt execution. This behavior is determined by the value of the stopExecution attribute. (The default is “Yes” to stop program execution.)

Note: The stopExecution attribute is the only attribute that is editable in a program halt component. All other attributes are set by the component itself.

The type of error that cause the program halt is stored to the type attribute. The Program Halt Component can trap Generic, Generic Math, Overflow, Underflow, or Zero Divide errors.

The dump attribute stores the list ID of the list that is otherwise displayed in the SAS Log when a program halt occurs. You can use this list to identify problems in your application. For example, you could implement a program halt component and override the error handling methods to send e-mail of the dump when the program halt occurs.

### Attributes

Public attributes specified for the Program Halt Component class are described here. See sashelp.fsp.Object.class for inherited attributes.

### Dictionary

**description**

*Returns the short description for the object*

Type: Character

Valid Values:

**dump**

*Returns a list containing the actual program halt*

Type: List

Valid Values:
entry

Returns the four-level name of the SAS/AF entry where the program halt occurred

Type: Character
Valid Values:

keyword

Returns the name of the method or function on which the program halt occurred. This attribute only applies for generic events.

Type: Character
Valid Values:

keywordType

Returns whether the keyword attribute contains a method name or a function name. If keywordType is blank, the program halt occurred on some other statement.

Type: Character
Valid Values:

lineNumber

Returns the line number in the entry where the program halt occurred

Type: Numeric
Valid Values:
stopExecution

Specifies whether to halt execution completely or continue executing the program: YES continues the program flow (default), NO stops execution completely

Type: Character
Valid Values: Yes, No

traceback

Returns a list that contains the same information that the SCL traceback function provides

Type: List
Valid Values:

type

Returns the type of error that occurred: GENERIC, GENERICMATH, OVERFLOW, UNDERFLOW, ZERODIVIDE or ATTRIBUTEERROR

Type: Character
Valid Values:

Methods

Methods specified for the Program Halt Component class are described here. See sashelp.fsp.Object.class for inherited methods.

_onAttributeError

Invoked when the attributeError event occurs
Program Halt Component

---

**Syntax**

```
objectName_onAttributeError();
```

---

**_onGeneric**

Invoked when the generic event occurs

**Syntax**

```
objectName_onGeneric();
```

---

**_onGenericMath**

Invoked when the genericMath event occurs

**Syntax**

```
objectName_onGenericMath();
```

---

**_onOverFlow**

Invoked when the overFlow event occurs

**Syntax**

```
objectName_onOverFlow();
```
Events

Events specified for the Program Halt Component class are described here.

attributeError
  Occurs as a result of an attribute error

generic
  Occurs as a result of a generic error

genericMath
  Occurs as a result of a genericMath error

overFlow
  Occurs as a result of an overflow error

underFlow
  Occurs as a result of an underflow error
zeroDivide

Occurs as a result of a zeroDivide error

## Event Handlers

Event handlers specified for the Program Halt Component class are described here.

<table>
<thead>
<tr>
<th>Event</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.attributeError</td>
<td>._onAttributeError</td>
<td>Executes when the attributeError event occurs</td>
</tr>
<tr>
<td>.generic</td>
<td>._onGeneric</td>
<td>Executes when the generic event occurs</td>
</tr>
<tr>
<td>.genericMath</td>
<td>._onGenericMath</td>
<td>Executes when the genericMath event occurs</td>
</tr>
<tr>
<td>.overFlow</td>
<td>._onOverFlow</td>
<td>Executes when the overFlow event occurs</td>
</tr>
<tr>
<td>.underFlow</td>
<td>._onUnderFlow</td>
<td>Executes when the underFlow event occurs</td>
</tr>
<tr>
<td>.zeroDivide</td>
<td>._onZeroDivide</td>
<td>Executes when the zeroDivide event occurs</td>
</tr>
</tbody>
</table>